Lenguajes y Compiladores

2015

Estructura de la materia a grandes rasgos:

Primera Parte: Lenguaje imperativo

Segunda Parte: Lenguaje aplicativo puro, y lenguaje aplicativo con referencias y asignación

Ejes de contenidos de la segunda parte

El Cálculo Lambda

Church, Kleene, Rosser (1930)

Motivación original: problemas de fundamentación de la matemática

El Cálculo Lambda

Church, Kleene, Rosser (1930)

Motivación original: problemas de fundamentación de la matemática

Como notación: se usa para generar expresión que denote una función, sin necesidad de dar nombre

Por ejemplo: $f(x) = x + y$ se puede escribir

$$f = \lambda x.\ x + y,$$

con lo cual uno se puede referir a la función f mediante la expresión $\lambda x.x + y$, sin necesidad de ponerle nombre.

Sintaxis abstracta del CL

El CL puro sólo contiene variables, aplicaciones y la notación lambda (llamada abstracción).

$$
\begin{array}{lll}
\langle exp \rangle & ::= & \langle var \rangle \mid \\
 & & \langle exp \rangle \, \langle exp \rangle \mid \\
 & & \lambda \, \langle var \rangle \, . \, \langle exp \rangle \quad \text{abstracción o expresión lambda}
\end{array}
$$

Sintaxis abstracta del CL

El CL puro sólo contiene variables, aplicaciones y la notación lambda (llamada abstracción).

$$\begin{array}{lll} \langle exp \rangle & ::= & \langle var \rangle \mid \\ & & \langle exp \rangle \, \langle exp \rangle \mid \\ & & \lambda \, \langle var \rangle \, . \, \langle exp \rangle \quad \text{abstracción o expresión lambda} \end{array}$$

La aplicación asocia a izquierda.

Por ejemplo,

$$\lambda x.(\lambda y.xyx)x$$

es lo mismo que

$$\lambda x.(\lambda y.(xy)x)x$$

Variables libres

$$
\begin{aligned}
FV\ v &= \{v\} \\
FV\ ee' &= FV\ e \cup FV\ e' \\
FV(\lambda v.e) &= FV\ e - \{v\}
\end{aligned}
$$

Operador sustitución

Conjunto de sustituciones: $\Delta = \langle var \rangle \rightarrow \langle exp \rangle$

Operador sustitución

Conjunto de sustituciones: $\Delta = \langle var \rangle \rightarrow \langle exp \rangle$

Operador sustitución: $\_/\_ \in \langle exp \rangle \times \Delta \rightarrow \langle exp \rangle$

Operador sustitución

Conjunto de sustituciones: $\Delta = \langle var \rangle \to \langle exp \rangle$

Operador sustitución: $\_/\_ \in \langle exp \rangle \times \Delta \to \langle exp \rangle$

$$\begin{aligned} v/\delta &= \delta v \\ (ee')/\delta &= (e/\delta)(e'/\delta) \end{aligned}$$

Operador sustitución

Conjunto de sustituciones: $\Delta = \langle var \rangle \rightarrow \langle exp \rangle$

Operador sustitución: $\_/\_ \in \langle exp \rangle \times \Delta \rightarrow \langle exp \rangle$

$$\begin{aligned} v/\delta &= \delta v \\ (ee')/\delta &= (e/\delta)(e'/\delta) \\ (\lambda v.e)/\delta &= \lambda v.\ e/[\delta | v : v_{new}] \end{aligned}$$

donde

$$v_{new} \notin \bigcup_{w \in FV(e)-\{v\}} FV(\delta\ w)$$

Conversión α

Renombre: Cambio en $\lambda v.e$ de la variable ligada v (y todas sus ocurrencias) por una variable v' que no ocurra libre en e:

$$\lambda v'.\ e/v \mapsto v'$$

donde $v' \notin FV(e)$.

Conversión α

Renombre: Cambio en $\lambda v.e$ de la variable ligada v (y todas sus ocurrencias) por una variable v' que no ocurra libre en e:

$$\lambda v'.\ e/v \mapsto v'$$

donde $v' \notin FV(e)$.

α-conversión: Si e' se obtiene a partir de e por 0 o más renombres de ocurrencias de subfrases. También se dice que e α-convierte a e'.

Conversión α

Renombre: Cambio en $\lambda v.e$ de la variable ligada v (y todas sus ocurrencias) por una variable v' que no ocurra libre en e:

$$\lambda v'.\ e/v \mapsto v'$$

donde $v' \notin FV(e)$.

α-conversión: Si e' se obtiene a partir de e por 0 o más renombres de ocurrencias de subfrases. También se dice que e α-convierte a e'.

Notación para expresiones α- convertibles: $e \equiv e'$

Contracción β

Redex: Es una expresión de la forma $(\lambda v.e)e'$

Contracción β

Redex: Es una expresión de la forma $(\lambda v.e)e'$

Contracción β de e_0: Reemplaza en e_0 una ocurrencia de un redex $(\lambda v.e)e'$ por su contracción $(e/v \mapsto e')$, y luego efectúa 0 o más renombres de cualquier subexpresión.

Contracción β

Redex: Es una expresión de la forma $(\lambda v.e)e'$

Contracción β de e_0: Reemplaza en e_0 una ocurrencia de un redex $(\lambda v.e)e'$ por su contracción $(e/v \mapsto e')$, y luego efectúa 0 o más renombres de cualquier subexpresión.

Notación: Si e_1 es el resultado de una contracción β de e_0, entonces escribimos

$$e_0 \to e_1$$

Formas normales y semántica operacional del CL

Forma normal: expresión sin redices.

Las formas normales representan la situación de "completa evaluación de todas las funciones".

Formas normales y semántica operacional del CL

Forma normal: expresión sin redices.

Las formas normales representan la situación de "completa evaluación de todas las funciones".

Por eso la semántica operacional del cálculo lambda consiste en efectuar contracciones β hasta obtener formas normales.

Ejemplo:

$$(\lambda x.\lambda y.(\lambda x.xy)(zx))(yz) \quad \rightarrow \quad (\lambda x.\lambda y.(zx)y)(yz)$$

Formas normales y semántica operacional del CL

Forma normal: expresión sin redices.

Las formas normales representan la situación de "completa evaluación de todas las funciones".

Por eso la semántica operacional del cálculo lambda consiste en efectuar contracciones β hasta obtener formas normales.

Ejemplo:

$$\begin{aligned}(\lambda x.\lambda y.(\lambda x.xy)(zx))(yz) &\rightarrow (\lambda x.\lambda y.(zx)y)(yz) \\ &\equiv (\lambda x.\lambda t.(zx)t)(yz)\end{aligned}$$

Formas normales y semántica operacional del CL

Forma normal: expresión sin redices.

Las formas normales representan la situación de "completa evaluación de todas las funciones".

Por eso la semántica operacional del cálculo lambda consiste en efectuar contracciones β hasta obtener formas normales.

Ejemplo:

$$\begin{aligned}(\lambda x.\lambda y.(\lambda x.xy)(zx))(yz) &\rightarrow (\lambda x.\lambda y.(zx)y)(yz)\\ &\equiv (\lambda x.\lambda t.(zx)t)(yz)\\ &\rightarrow \lambda t.z(yz)t\end{aligned}$$

No toda expresión tiene una forma normal

Sea

$$\Delta = \lambda x.xx$$

entonces $\Delta\Delta$ no tiene forma normal:

$$(\lambda x.xx)(\lambda x.xx) \quad \rightarrow \quad (\lambda x.xx)(\lambda x.xx)$$

No toda expresión tiene una forma normal

Sea

$$\Delta = \lambda x.xx$$

entonces $\Delta\Delta$ no tiene forma normal:

$$\begin{aligned}(\lambda x.xx)(\lambda x.xx) &\rightarrow (\lambda x.xx)(\lambda x.xx)\\ &\rightarrow (\lambda x.xx)(\lambda x.xx)\\ &\vdots\end{aligned}$$

Relación "reduce" $\rightarrow^*$

$\rightarrow^*$ denota la clausura transitiva y refexiva de $\rightarrow$

(o sea, aplicar $\rightarrow$ cero o más veces)

Relación "reduce" $\rightarrow^*$

$\rightarrow^*$ denota la clausura transitiva y refexiva de $\rightarrow$

(o sea, aplicar $\rightarrow$ cero o más veces)

Por ejemplo, no existe n forma normal tal que $\Delta\Delta \rightarrow^* n$

Relación "reduce" $\rightarrow^*$

$\rightarrow^*$ denota la clausura transitiva y refexiva de $\rightarrow$

(o sea, aplicar $\rightarrow$ cero o más veces)

Por ejemplo, no existe n forma normal tal que $\Delta\Delta \rightarrow^* n$

Formalmente:

$e \rightarrow^* e'$ si y sólo si existen $e_0, ..., e_n$ (con $n \geq 0$) tales que

$$e = e_0 \rightarrow e_1 \rightarrow ... \rightarrow e_n = e'$$

Notar que si $n = 0$ entonces $e = e'$

Si existe forma normal, es única

(Salvo conversión α)

Es consecuencia inmediata de:

Teorema de Church-Rosser Si $e \to^* e_0$ y $e \to^* e_1$ entonces existe e' tal que $e_0 \to^* e'$ y $e_1 \to^* e'$.

Regla η

Un η-redex es una expresión de la forma $\lambda v.ev$, donde $v \notin FV\ e$

Regla η

Un η-redex es una expresión de la forma $\lambda v.ev$, donde $v \notin FV\ e$

Regla η:

$$\frac{}{\lambda v.ev\ \rightarrow\ e} \qquad \text{(si } v \notin FV\ e\text{)}$$

Evaluación

$\rightarrow^*$ no representa adecuadamente la ejecución de los lenguajes aplicativos.

Evaluación

$\rightarrow^*$ no representa adecuadamente la ejecución de los lenguajes aplicativos. En estos:

1) sólo se evalúan expresiones cerradas

Evaluación

$\rightarrow^*$ no representa adecuadamente la ejecución de los lenguajes aplicativos. En estos:

1) sólo se evalúan expresiones cerradas
2) es determinística

Evaluación

$\rightarrow^*$ no representa adecuadamente la ejecución de los lenguajes aplicativos. En estos:

1) sólo se evalúan expresiones cerradas
2) es determinística
3) no busca formas normales sino formas canónicas

Evaluación

$\rightarrow^*$ no representa adecuadamente la ejecución de los lenguajes aplicativos. En estos:

1) sólo se evalúan expresiones cerradas
2) es determinística
3) no busca formas normales sino formas canónicas

Vamos a estudiar:

Evaluación (en orden) normal: lenguajes funcionales lazy (Haskell)

Evaluación eager o estricta: lenguajes estrictos (ML).

Formas canónicas

La evaluación busca una forma canónica, las mismas juegan el rol de ser "valores" de expresiones.

Formas canónicas

La evaluación busca una forma canónica, las mismas juegan el rol de ser "valores" de expresiones.

La noción de forma canónica depende de la definición de evaluación. Se define una noción de forma canónica para la evaluación normal, y otra para la evaluación eager.

Formas canónicas

La evaluación busca una forma canónica, las mismas juegan el rol de ser "valores" de expresiones.

La noción de forma canónica depende de la definición de evaluación. Se define una noción de forma canónica para la evaluación normal, y otra para la evaluación eager.

En el caso del cálculo lambda coinciden: **son las abstracciones**

Formas canónicas vs. formas normales

Propiedad: Una aplicación cerrada no puede ser forma normal.

Formas canónicas vs. formas normales

Propiedad: Una aplicación cerrada no puede ser forma normal.

Corolario: una expresión cerrada que es forma normal es también forma canónica.

El recíproco no vale.

¿Por qué conformarse con una forma canónica en vez de continuar ejecutando hasta obtener una forma normal?

¿Por qué conformarse con una forma canónica en vez de continuar ejecutando hasta obtener una forma normal?

Sólo tiene sentido evaluar expresiones cerradas.

¿Por qué conformarse con una forma canónica en vez de continuar ejecutando hasta obtener una forma normal?

Sólo tiene sentido evaluar expresiones cerradas.

Una vez que se alcanzó una abstracción $\lambda v.e$, continuar evaluando implicaría evaluar e que puede no ser una expresión cerrada, puede contener a la variable v.

semántica natural o big-step

En este tipo de semántica, uno no describe un paso de ejecución, sino directamente una relación entre los términos y sus valores (que también son términos, son formas canónicas).

semántica natural o big-step

En este tipo de semántica, uno no describe un paso de ejecución, sino directamente una relación entre los términos y sus valores (que también son términos, son formas canónicas).

Llamaremos $\Rightarrow$ a esta relación.

Evaluación Normal

Reglas para $\Rightarrow_N$

Regla para las formas canónicas

$$\frac{}{\lambda v.e \Rightarrow_N \lambda v.e}$$

Evaluación Normal

Reglas para $\Rightarrow_N$

Regla para las formas canónicas

$$\frac{}{\lambda v.e \ \Rightarrow_N \ \lambda v.e}$$

Regla para la aplicación

$$\frac{e \ \Rightarrow_N \ \lambda v.e_0 \qquad (e_0/v \mapsto e') \ \Rightarrow_N \ z}{ee' \ \Rightarrow_N \ z}$$

Evaluación Eager

Reglas para $\Rightarrow_E$

Regla para las formas canónicas

$$\frac{}{\lambda v.e \Rightarrow_E \lambda v.e}$$

Evaluación Eager

Reglas para $\Rightarrow_E$

Regla para las formas canónicas

$$\frac{}{\lambda v.e \ \Rightarrow_E \ \lambda v.e}$$

Regla para la aplicación

$$\frac{e \ \Rightarrow_E \ \lambda v.e_0 \qquad e' \ \Rightarrow_E \ z' \qquad (e_0/v \mapsto z') \ \Rightarrow_E \ z}{ee' \ \Rightarrow_E \ z}$$

Semántica Denotacional del Cálculo Lambda

Surge con Church, trantado de construir un sistema formal completo como fundamentación última de la matemática. En 1934 Kleene and Rosser publicaron una implementación de la paradoja de Richard,y comienza a ser usado para estudiar la computabilidad, culminando en la respuesta negativa al problema de la parada (Turing 1936).

Semántica Denotacional del Cálculo Lambda

Requisito inicial para su semántica:

un conjunto C tal que C es isomorfo a $C \to C$

Semántica Denotacional del Cálculo Lambda

Requisito inicial para su semántica:

un conjunto C tal que C es isomorfo a $C \to C$

Paradoja: Sea f cualquier función, y definamos $px = f(xx)$

Entonces pp es punto fijo de f.

Semántica Denotacional del Cálculo Lambda

Requisito inicial para su semántica:

un conjunto C tal que C es isomorfo a $C \to C$

Paradoja: Sea f cualquier función, y definamos $px = f(xx)$

Entonces pp es punto fijo de f.

Scott-Strachey 1972-1981 Oxford Definen la semántica utilizando dominios:

$$D_\infty \simeq D_\infty \to D_\infty$$

Semántica Denotacional del Cálculo Lambda

Asumimos la existencia de un dominio D_∞, junto con isomorfismos:

$$\phi \in D_\infty \to (D_\infty \to D_\infty)$$

$$\psi \in (D_\infty \to D_\infty) \to D_\infty$$

$$\phi \circ \psi = Id_{D_\infty \to D_\infty}$$

$$\psi \circ \phi = Id_{D_\infty}$$

Semántica Denotacional del Cálculo Lambda

Ambientes (Entornos): $Env = (\langle var \rangle \rightarrow D_\infty)$

Notación: $\eta \in Env$ será un ambiente.

Semántica Denotacional del Cálculo Lambda

Ambientes (Entornos): $Env = (\langle var \rangle \to D_\infty)$

Notación: $\eta \in Env$ será un ambiente.

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D_\infty$

Semántica Denotacional del Cálculo Lambda

Ambientes (Entornos): $Env = (\langle var \rangle \to D_\infty)$

Notación: $\eta \in Env$ será un ambiente.

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D_\infty$

Ecuaciones semánticas:

$[\![v]\!]\eta = \eta v$

Semántica Denotacional del Cálculo Lambda

Ambientes (Entornos): $Env = (\langle var \rangle \to D_\infty)$

Notación: $\eta \in Env$ será un ambiente.

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D_\infty$

Ecuaciones semánticas:

$[\![v]\!]\eta = \eta v$

$[\![e_0\, e_1]\!]\eta = \phi([\![e_0]\!]\eta)\ [\![e_1]\!]\eta$

Semántica Denotacional del Cálculo Lambda

Ambientes (Entornos): $Env = (\langle var \rangle \to D_\infty)$

Notación: $\eta \in Env$ será un ambiente.

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D_\infty$

Ecuaciones semánticas:

$$[\![v]\!]\eta = \eta v$$

$$[\![e_0\, e_1]\!]\eta = \phi([\![e_0]\!]\eta)\ [\![e_1]\!]\eta$$

$$[\![\lambda v.e]\!]\eta = \psi(\lambda d \in D_\infty.[\![e]\!][\eta|v : d])$$

Propiedades de la semántica del CL

Teorema de Coincidencia:
Si $\eta w = \eta' w$ para todo $w \in FV\ e$, entonces $[\![e]\!]\eta = [\![e]\!]\eta'$.

Propiedades de la semántica del CL

Teorema de Coincidencia:
Si $\eta w = \eta' w$ para todo $w \in FV\ e$, entonces $[\![e]\!]\eta = [\![e]\!]\eta'$.

Teorema de Renombre: Si $v_{new} \notin FV\ e - \{v\}$,
entonces $[\![\lambda v_{new}.(e/v \mapsto v_{new})]\!] = [\![\lambda v.e]\!]$.

Propiedades de la semántica del CL

Teorema de Coincidencia:
Si $\eta w = \eta' w$ para todo $w \in FV\ e$, entonces $[\![e]\!]\eta = [\![e]\!]\eta'$.

Teorema de Renombre: Si $v_{new} \notin FV\ e - \{v\}$,
entonces $[\![\lambda v_{new}.(e/v \mapsto v_{new})]\!] = [\![\lambda v.e]\!]$.

Sustituciones: $\Delta = \langle var \rangle \rightarrow \langle exp \rangle$

Propiedades de la semántica del CL

Teorema de Coincidencia:
Si $\eta w = \eta' w$ para todo $w \in FV\ e$, entonces $[\![e]\!]\eta = [\![e]\!]\eta'$.

Teorema de Renombre: Si $v_{new} \notin FV\ e - \{v\}$,
entonces $[\![\lambda v_{new}.(e/v \mapsto v_{new})]\!] = [\![\lambda v.e]\!]$.

Sustituciones: $\Delta = \langle var \rangle \to \langle exp \rangle$

Teorema de Sustitución: Si $[\![\delta w]\!]\eta = \eta' w$ para todo $w \in FV\ e$,
entonces $[\![e/\delta]\!]\eta = [\![e]\!]\eta'$.

Reglas β y η

Son válidas en el Cálculo Lambda:

$[\![(\lambda v.e)e']\!]\eta = [\![e/v \mapsto e']\!]\eta$

$[\![\lambda v.ev]\!]\eta = [\![e]\!]\eta$, si $v \notin FVe$

Problema

La semántica denotacional dada no representa la evaluación de los lenguajes funcionales

Problema

La semántica denotacional dada no representa la evaluación de los lenguajes funcionales

Observar que $\lambda x.e$ se evalúa como función en cualquier modalidad de evaluación (tanto eager como normal).

Problema

La semántica denotacional dada no representa la evaluación de los lenguajes funcionales

Observar que $\lambda x.e$ se evalúa como función en cualquier modalidad de evaluación (tanto eager como normal).

Pero $[\![\lambda v.\Delta\Delta]\!] = \perp$

Semántica Denotacional Normal

Hay que distinguir las semántica de $\lambda v.\Delta\Delta$ y $\Delta\Delta$

Semántica Denotacional Normal

Hay que distinguir las semántica de $\lambda v.\Delta\Delta$ y $\Delta\Delta$

Los "valores" (conjunto V), representan a las formas canónicas:

$$V \approx D \to D$$

Semántica Denotacional Normal

Hay que distinguir las semántica de $\lambda v.\Delta\Delta$ y $\Delta\Delta$

Los "valores" (conjunto V), representan a las formas canónicas:

$$V \approx D \to D$$

Definimos el conjunto de resultados posibles como :

$$D = V_{\perp}$$

Asumimos la existencia de un dominio D, junto con isomorfismos:

$$\phi \in V \to (D \to D)$$

$$\psi \in (D \to D) \to V$$

$$\phi \circ \psi = Id_{D \to D}$$

$$\psi \circ \phi = Id_V$$

Notación: $\iota_\perp \in V \to D$

Semántica Denotacional del Cálculo Lambda Normal

Dominio Semántico: $D = V_{\perp}$ $\quad V \approx (D \to D)$

Semántica Denotacional del Cálculo Lambda Normal

Dominio Semántico: $D = V_{\perp}$ $\quad V \approx (D \to D)$

Ambientes: $Env = (\langle var \rangle \to D)$

Semántica Denotacional del Cálculo Lambda Normal

Dominio Semántico: $D = V_{\perp} \qquad V \approx (D \to D)$

Ambientes: $Env = (\langle var \rangle \to D)$

Función semántica: $[\![\_]\!] \in \quad \langle exp \rangle \to Env \to D$

Semántica Denotacional del Cálculo Lambda Normal

Dominio Semántico: $D = V_\perp \qquad V \approx (D \to D)$

Ambientes: $Env = (\langle var \rangle \to D)$

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D$

Ecuaciones semánticas:

$[\![v]\!]\eta = \eta v$

Semántica Denotacional del Cálculo Lambda Normal

Dominio Semántico: $D = V_\perp \qquad V \approx (D \to D)$

Ambientes: $Env = (\langle var \rangle \to D)$

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D$

Ecuaciones semánticas:

$[\![v]\!]\eta = \eta v$

$[\![e_0\, e_1]\!]\eta = \phi([\![e_0]\!]\eta) \perp\!\!\!\perp [\![e_1]\!]\eta$

Semántica Denotacional del Cálculo Lambda Normal

Dominio Semántico: $D = V_\perp \qquad V \approx (D \to D)$

Ambientes: $Env = (\langle var \rangle \to D)$

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D$

Ecuaciones semánticas:

$$[\![v]\!]\eta = \eta v$$

$$[\![e_0 e_1]\!]\eta = \phi([\![e_0]\!]\eta)_{\perp\!\!\perp}[\![e_1]\!]\eta$$

$$[\![\lambda v.e]\!]\eta = \iota_\perp \circ \psi\,(\lambda d \in D.[\![e]\!][\eta | v : d])$$

¿Cuáles son todas las posibilidades para que $e_0 e_1$ tenga semántica $\perp$

Notar que $\phi_{\perp\!\!\perp} \perp \;=\; \perp_{D \to D}$.

¿Cuáles son todas las posibilidades para que $e_0 e_1$ tenga semántica $\perp$?

Notar que $\phi_{\perp\!\!\perp} \perp \;=\; \perp_{D \to D}$.

Si bien la semántica denotacional no expresa un orden de evaluación ya que no es operacional, establece indirectamente el orden natural en que debe evaluarse.

¿Cuáles son todas las posibilidades para que $e_0 e_1$ tenga semántica $\perp$?

Notar que $\phi_{\perp\!\!\perp} \perp \; = \; \perp_{D \to D}$.

Si bien la semántica denotacional no expresa un orden de evaluación ya que no es operacional, establece indirectamente el orden natural en que debe evaluarse.

Por más que $[\![e_1]\!]\eta = \perp$, no necesariamente $[\![e_0 e_1]\!]\eta = \perp$. Eso indica que e_1 no necesariamente debe evaluarse para evaluarse $e_0 e_1$.

Propiedades de la Semántica Denotacional Normal

Los teoremas que vimos antes siguen valiendo.

Propiedades de la Semántica Denotacional Normal

Los teoremas que vimos antes siguen valiendo.

También vale la regla β, que utiliza la igualdad:

$$\phi_{\perp\!\!\perp} \circ (\iota_{\perp} \circ \psi) = Id_{D \to D}$$

Propiedades de la Semántica Denotacional Normal

Los teoremas que vimos antes siguen valiendo.

También vale la regla β, que utiliza la igualdad:

$$\phi_{\perp\!\!\perp} \circ (\iota_{\perp} \circ \psi) = Id_{D \to D}$$

No vale la regla η.

Modalidad EAGER

La semántica denotacional del cálculo Lambda, ni la normal representan la evaluación de los lenguajes funcionales eager

Por ejemplo: $(\lambda x.\lambda y.y)(\Delta\Delta)$

La aplicación se efectúa cuando el operando se haya evaluado. Desde el punto de vista semántico, las funciones toman valores solamente.

Semántica Denotacional Eager

Los "valores" (conjunto V), representan a las formas canónicas, pero ahora son funciones que sólo toman valores:

$$V \approx V \to D$$

Semántica Denotacional Eager

Los "valores" (conjunto V), representan a las formas canónicas, pero ahora son funciones que sólo toman valores:

$$V \approx V \to D$$

Definimos el conjunto de resultados posibles como :

$$D = V_{\perp}$$

Asumimos la existencia de un dominio D, junto con isomorfismos:

$$\phi \in V \to (V \to D)$$

$$\psi \in (V \to D) \to V$$

$$\phi \circ \psi = Id_{V \to D}$$

$$\psi \circ \phi = Id_V$$

Notación: $\iota_\perp \in V \to D$

Semántica Denotacional del Cálculo Lambda Eager

Dominio Semántico: $D = V_{\perp}$ $\quad V \approx (V \to D)$

Ambientes: $Env = (\langle var \rangle \to V)$

Semántica Denotacional del Cálculo Lambda Eager

Dominio Semántico: $D = V_{\perp} \qquad V \approx (V \to D)$

Ambientes: $Env = (\langle var \rangle \to V)$

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D$

Semántica Denotacional del Cálculo Lambda Eager

Dominio Semántico: $D = V_\perp \qquad V \approx (V \to D)$

Ambientes: $Env = (\langle var \rangle \to V)$

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D$

Ecuaciones semánticas:

$[\![v]\!]\eta = \iota_\perp(\eta v)$

Semántica Denotacional del Cálculo Lambda Eager

Dominio Semántico: $D = V_\perp \qquad V \approx (V \to D)$

Ambientes: $Env = (\langle var \rangle \to V)$

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D$

Ecuaciones semánticas:

$[\![v]\!]\eta = \iota_\perp(\eta v)$

$[\![e_0\, e_1]\!]\eta = \phi_{\perp\!\!\perp}([\![e_0]\!]\eta)_{\perp\!\!\perp}[\![e_1]\!]\eta$

Semántica Denotacional del Cálculo Lambda Eager

Dominio Semántico: $D = V_\perp \qquad V \approx (V \to D)$

Ambientes: $Env = (\langle var \rangle \to V)$

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D$

Ecuaciones semánticas:

$$[\![v]\!]\eta = \iota_\perp(\eta v)$$

$$[\![e_0\, e_1]\!]\eta = \phi_{\perp\!\!\perp}([\![e_0]\!]\eta)_{\perp\!\!\perp}[\![e_1]\!]\eta$$

$$[\![\lambda v.e]\!]\eta = \iota_\perp \circ \psi\,(\lambda z \in V.[\![e]\!][\eta | v : z])$$

Propiedades de la Semántica Denotacional Eager

Los teoremas que vimos antes siguen valiendo.

Propiedades de la Semántica Denotacional Eager

Los teoremas que vimos antes siguen valiendo.

Ya no vale la regla β.

$$[\![(\lambda v.e)e']\!]\eta = (\lambda z \in V.[\![e]\!][\eta|v:z])_{\perp\!\!\!\perp}[\![e']\!]\eta$$

Considerar: $[\![e']\!]\eta = \perp$, v, no ocurre en e, y $[\![e]\!]\eta \neq \perp$

Entonces $[\![(\lambda v.e)e']\!]\eta = \perp$ y $[\![e/v \mapsto e']\!]\eta \neq \perp$

No vale la regla η.